1 Introduction

Qu'est ce qu'un programme ?

C'est une suite d'instructions. On utilise souvent la méthapore de la recette de cuisine, avec ses ingrédients (=données) et les étapes (=instructions).

Voici différents paradigmes de programmation (=manière d'approcher la programmation et de concevoir les solutions aux problèmes donnés) :

Langages classiques (procéduraux)	Langages fonctionnels	Langages logiques
C, Pascal, FORTRAN, COBOL	LISP, CAML, Lambda-calculs	PROLOG

1.1 Autre paradigme : les langages objets

- *SIMULA* (65-70) : **Les classes**
  - Principe 1 : on raccroche à la structure de données (champs), les codes qui peuvent les manipuler (méthodes)
- *SMALLTALK* (70-75) : **L'héritage**
  - A. Kay (souris, interfaces graphiques)
  - Principe 2 : on hiérarchise les classes
- En dehors des labos :
  - 80 : SMALLTALK
  - EIFFEL, LISPOBJECT, PASCALOBJECT, C++, JAVA

1.2 Langages compilés vs interprétés

- Un compilateur traduit un programme (=source) en un fichier exécutable par le processeur (généralement en langage assembleur).
- Un interpréteur traduit et exécute ligne par ligne.
- Ainsi, un interpréteur gagne en souplesse (il peut éviter une ligne qui provoque une erreur), mais perd en rapidité (par exemple, dans une boucle, où il re-traduit les lignes à chaque tour).

2 Principe de la programmation objet

Les différences avec la programmation classique :

- **La programmation classique :**
  - Les procédures agissent sur les données
    1. on a une première idée des procédures
    2. on définit les variables (données)
    3. on écrit les procédures
  - une exécution = appels aux procédures dans un ordre, en donnant les données nécessaires
- **POO :**
  - On pense concept
  - Programmer : on raccroche à la structure de données (champs), les codes qui peuvent les manipuler (méthodes)
  - Exécuter : une seule entité homogène : l'objet (aspect données et activations)

2.1 Les concepts de base

La modélisation amène à concevoir mes classes (codes et données), les objets sont activités par envois de messages lors l'exécution.

2.1.1 Classes et instances

Définition 1 *Une classe est un moule (ou matrice) à partir duquel sont générés les objets.*

Un objet généré à partir d'une classe C est une instance de C.

Une classe a un nom (le nom du concept) ; elle est constituée d'un ensemble de champs (attributs, variables d'instances) qui décrivent la structure des objets, et d'opérations, appelées méthodes, qui leur sont applicables.

En LOLO (∼SMALLTALK)

```
classe Compte
 champs credit : Nbe
     debit : Nbe
FinCompte
```

En JAVA

```
class Compte {
 int credit;
 int debit;
}
```

Création d'un objet :

- Un objet est créé par appel à un constructeur (new) appliqué à sa classe. Le résultat de la création est un objet qui peut être placé dans une variable, un pointeur sur l'objet.
- Rappel : l'objet créé est dit *instance* de la classe qui l'a créé.

En LOLO

```
C <- Compte<=new()
```

En JAVA

```
Compte C = new Compte();
```

2.1.2 Méthodes et envois de messages

- La partie dynamique des objets est assurée par l'envoi de messages
- Instruction à un objet : Envoyer un message à un objet c'est lui dire le comportement qu'il doit adopter, c'est à dire la méthode qu'il doit s'appliquer.

Définition 2 *Une méthode est une description, au niveau d'une classe, d'un ensemble d'instructions.*

La méthode contient une liste de paramètres formels ; elle peut s'appliquer aux objets instances de la classe.

Un message est contitué du nom de la méthode (m) et d'une liste de paramètres effectifs.

- Lorsqu'un objet O reçoit un message M, il s'applique la méthode donnée dans M ; celle-ci étant définie dans la classe C de l'objet O

En UML (Langage de Modélisation Unifié)

Nomclasse
Champs ...
Méthodes ...

On appelle cela un diagramme de classe.

En LOLO

```
Classe Compte
 Champs ...
     ...
 Methode deposer(V : Nbre)
  credit <- credit + V
 Methode retirer(V : Nbre)
  credit <- credit - V
 Methode VoirSolde : Nbre
  Return credit - debit
FinCompte

Dupont <- Compte<=new()
Dupont<=deposer(50)
Dupont<=AvoirSolde()
```

En JAVA

```
Class Compte {
int ...;
...
public void deposer(int V){credit += V;}
public void retirer(int V){credit -= V;}
public int AvoirSolde(){return cerdit - debit}
}

compte Dupont = new Compte();
Dupont.deposer(100);
```

2.1.3 L'héritage

- Méthode :
  1. Lors de la conception, on regroupe dans une classe commune les caractéristiques (ou membres), c'est à dire les champs et méthodes, communes de classes se ressemblant
  2. On hiérarchise les classes avec le lien **« est-une-sorte-de »**

Définition 3 *Une sous classe SC d'une classe C, est une classe sorte de C.*

SC possède par défaut les caractéristiques de C.

On dit que SC est une spécialisation de C, et que C est une généralisation de SC.

- La hiérarchie permet ainsi de factoriser l'écriture (et MAJ=Mise A Jour) de caractéristiques de plusieurs (sous) classes.

- On dit qu'une classe (ou objet) hérite des caractéristiques de ses sur-classes.

2.2 L'interprétation des messages

L'effet d'un message dépend du receveur (l'objet) et de l'héritage (hiérarchie).

2.2.1 L'effet dépend du receveur

- Un objet ne peut recevoir un message que si le sélecteur appartient à la classe de l'objet ou à une de ses sur-classes
- Il est possible d'envoyer deux messages avec le même sélecteur à deux objets différents qui ne sont pas de la même classe. Chaque classe implémentant sa propre méthode

2.2.2 L'effet dépend de la hiérarchie d'héritage

- Règle : Lorsqu'un objet reçoit un message de sélecteur M, il recherche d'abord la méthode de nom M dans sa classe, puis s'il ne la trouve pas, il recherche en remontant dans la hiérarchie des classes héritées.

2.2.3 La composition des messages

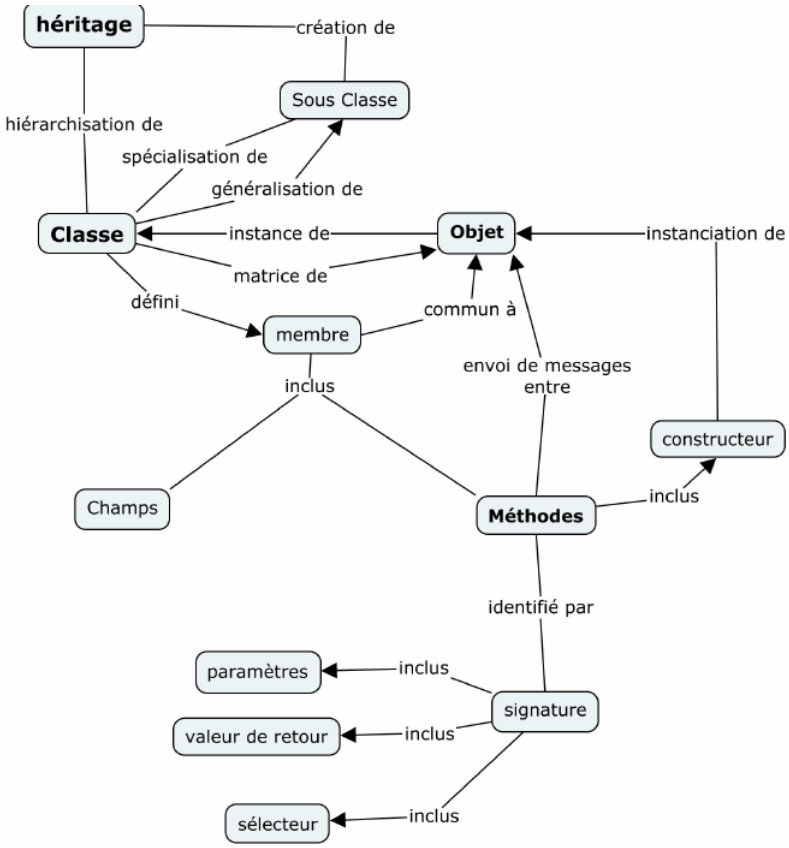
Définition 4 *Lorsqu'un message envoyé à un objet retourne un objet qui lui même reçoit un message, on dit qu'il y a composition de messages.*

2.2.4 Les messages à soi-même

- Les objets ont besoin de s'envoyer des messages à eux-même
- Il existe une « variable » pour cela appelée (*self* en SMALLTALK , *this* en JAVA et C++)

2.2.5 Les méthodes quasi-génériques

- Grâce a l'héritage et self (this)
- On peut définir des méthodes attachées à une classe indépendantes de leurs réalisations dans des sous-classes


héritage
création de
Sous Classe
hiérarchisation de
spécialisation de
généralisation de
Classe
instance de
Objet
instanciation de
matrice de
défini
commun à
membre
envoi de messages
entre
inclus
constructeur
Champs
Méthodes
inclus
identifié par
paramètres
inclus
signature
valeur de retour
inclus
sélecteur
inclus